		Количество строк в файле								
		5	10	100	1000	10000	100000	1000000	10000000	100000000
Сортировки	bubble	00:00:00:00.808	00:00:00:00.697	00:00:00:00.849	00:00:00:03.940	00:00:06:56.484	n\a	n\a	n\a	n\a
	insertion	00:00:00:00.662	00:00:00:00.941	00:00:00:01.163	00:00:00:04.103	00:00:06:33.328	n\a	n\a	n\a	n\a
	merge	00:00:00:00.664	00:00:00:00.692	00:00:00:00.770	00:00:00:00.978	00:00:00:05.011	00:00:00:45.146	00:00:08:09.439	00:01:20:59.026	n\a
	quick	00:00:00:00.880	00:00:00:00.758	00:00:00:00.879	00:00:00:00.888	00:00:00:03.887	00:00:00:31.321	00:00:04:19.453	00:00:53:47.961	n\a
	heap	00:00:00:00.685	00:00:00:00.663	00:00:00:00.682	00:00:00:01.035	00:00:00:04.946	00:00:00:49.605	00:00:08:06.757	00:01:22:37.485	n\a

Выводы: Для эксперимента был написан генератор строк, состоящих из заглавных и строчных латинских букв и цифр, длины 29. При больших объемах данных становится очевидна разница во времени работы сортировок с асимптотикой O(n^2) (bubble, insertion) по сравнению с сортировками с асимптотикой O(nlogn). Так же заметно увеличение времени работы merge, который использует дополнительную память. У heap sort очень большая константа, поэтому при одинаковой асимптотике с quick sort фактическое время работы первого алгоритма больше, чем второго. При N = 100000000 не удалось дождаться результата ни одной из сортировок предположительно из-за переполнения стека. Самой быстрой сортировкой на всех объемах данных оказалась quick sort.

Лог Валгринда:

```
==3256==
==3256== HEAP SUMMARY:
==3256==     in use at exit: 0 bytes in 0 blocks
==3256==   total heap usage: 314 allocs, 314 frees, 10,702 bytes allocated
==3256==
==3256== All heap blocks were freed -- no leaks are possible
==3256==
```